Paper to be submitted to Special Issue of Journal of Grid Generation, or to Computers and Fluids, or to AIAA Journal

OVERGRID - A Unified Overset Grid Generation Graphical Interface

William M. Chan *
NASA Ames Research Center
Mail Stop T27B-2
Moffett Field, California 94035-1000

Abstract

This paper presents a unified graphical interface and gridding strategy for performing overset grid generation. The interface called OVERGRID has been specifically designed to follow an efficient overset gridding strategy, and contains general grid manipulation capabilities as well as modules that are specifically suited for overset grids. General grid utilities include functions for grid redistribution, smoothing, concatenation, extraction, extrapolation, projection, and many others. Modules specially tailored for overset grids include a seam curve extractor, hyperbolic and algebraic surface grid generators, a hyperbolic volume grid generator, and a Cartesian box grid generator. Grid visualization is achieved using OpenGL while widgets are constructed with Tcl/Tk. The software is portable between various platforms from UNIX workstations to personal computers.

1. Introduction

In recent years, overset grid methods[1] have been successfully used to perform both static and moving-body flow computations for many complex configurations. These computations have contributed to the design and analysis of aerospace and marine vehicles at government laboratories[2-9] as well as in industry.[10-13] A key element in such work is to reduce the time required to obtain a solution and thus improve the design cycle time. The computational analysis process typically consists of grid generation, flow calculation, and post-processing. Despite recent advances in tools development, it is commonly agreed among practitioners of overset methods that grid generation remains the most labor intensive step.

Two main approaches are being pursued to reduce overset grid generation time. The first is to develop automated algorithms and software tools that reduce or eliminate the user's input at each step of the grid generation process. Examples of recent efforts are given in Ref. 14 and Ref. 15. For many years, users of overset methods have had to rely on grid generation tools designed for patched abutting grids.[16,17] More recently, software tools based on hyperbolic methods that are more suitable for efficient overset grid generation have been developed but these have existed as individual batch mode codes.[18-20] The second approach is to incorporate all essential and robust overset grid generation tools under a single environment which includes a graphical user interface (GUI). This paper describes one of the first efforts ever made towards this goal.

*Senior Research Scientist, MCAT, Inc.

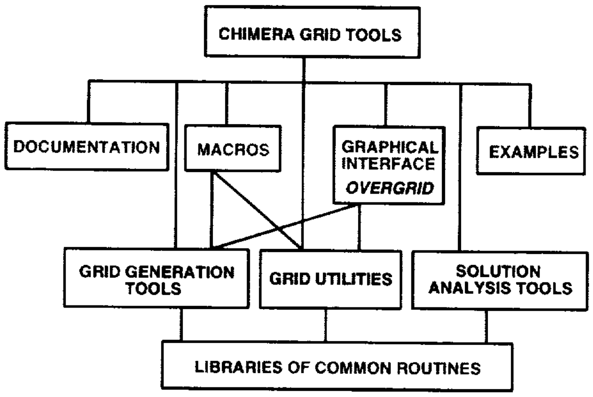

Figure 1: Structure of Chimera Grid Tools.

2. Overview of Grid Generation Interface

2.1. Software Structure and Design

The unified overset grid generation environment consists of a number of individual grid generation and manipulation modules that run in batch mode, and a graphical user interface OVERGRID that acts as a portal to the different modules as well as a visualization tool. The above collection of software is contained in a package called Chimera Grid Tools which also encompasses a number of solution analysis (post-processing) utilities. This paper will focus on the OVERGRID graphical interface and its implementation of the grid generation modules. A diagram showing the structure of Chimera Grid Tools is shown in Figure 1.

From OVERGRID, the individual grid modules are called as batch processes. By keeping each grid module separate from the graphical interface, a sequence of grid operations can be recorded in a script and reproduced easily without manual intervention. OVERGRID offers a script generation capability which automatically records all of the user's actions that affect the current contents in OVERGRID's memory into a script. If a simple modification in one of the steps is desired (e.g., alter the number of points in a grid), the script can be edited and re-run in batch mode to reproduce all the steps rapidly.

Another advantage of calling batch mode modules from OVERGRID is the handling of single and double precision data. OVERGRID keeps a single precision version of the current data in memory for display purposes, but performs operations instructed by the user on a copy of the data on disk. A preference setting in OVERGRID specifies a directory where executables of the batch mode modules are located. These executables will be used to operate on the copy of the data on disk. If the data is double precision, a directory of double precision executables is specified, and similarly for single precision. The above procedure allows OVERGRID to operate on single or double precision data without the need for code re-compilation. It is assumed that the single precision data used for display is always sufficient.

At any point during the execution of OVERGRID, the current contents in memory are stored on disk in an original state file. After an action is performed by the user, a new state file is created. Before the next action, the new state file is copied into the original state file. This procedure allows an easy way to undo the most recent action.

OVERGRID is primarily written in C and Tcl/Tk,[21,22] but all of the individual batch mode grid modules are written in FORTRAN for efficient execution speed. Rendering of objects is accomplished with OpenGL calls from the C main program, while the interface widgets (buttons, entries, and others) are built with Tcl/Tk. The ease of use and conciseness of Tcl/Tk have been found to be enormously valuable in rapidly creating the desired professional look and feel of the GUI. Moreover, the quick learning time allows the developers of grid generation codes to build the graphical interface themselves. This usually results in a much more practical and user-friendly graphical interface than one built by GUI experts with limited grid generation experience. OpenGL rendering from a Tcl/Tk window is made possible by the Togl widget *(http://www.cs.umd.edu/~bederson/Togl.html)*.

2.2. Data Formats and Main Windows

OVERGRID acts as focal point from which various grid modules can be accessed. Moreover, it serves as a visualization tool for the geometry and grids as they are read, created, modified and improved during grid generation. For the rest of this paper, the word 'entity' will be used to denote any object stored in OVERGRID as part of a surface geometry description (surfaces or curves) or a grid (volumes, surfaces, or curves). Entities in OVERGRID are stored as discrete points and a J/K/L convention is employed to denote the index directions. Currently, OVERGRID can only read and write entities in various PLOT3D[23] formats - unformatted or formatted, single or multiple zone, with or without iblanks, single or double precision. The above types for an input file are automatically determined by OVERGRID. Practical geometry surface definitions are frequently represented by Non-Uniform Rational B-Spline (NURBS) surfaces, triangles, solids, or panel networks. OVERGRID's current restriction of allowing only PLOT3D data formats implies it can only read surface geometry defined by multiple panel networks (each panel network is a rectangular array of points). Another software package such as GRIDGEN[16] or ICEMCFD[17] is usually used to convert other data formats into panel networks prior to using OVERGRID. Future plans for OVERGRID to read other data formats are given in Section 5.

On execution, OVERGRID will bring up the following four main windows shown in Figure 2.

(1) The Display window contains a graphical display of the entities currently in memory.
(2) The Controls window contains widgets for setting various display options, resetting views, showing information on the number of volume, surface and curve entities currently in memory.
(3) The Main window contains widgets for input and output of entities, script creation, access to the various grid modules, and general on-line help. Further detailed on-line help is also available under the window for the individual modules.
(4) The Selection window contains widgets for performing selection of entities, blanking/unblanking of entities from view, and deletion of entities. A more

direct entity selection mechanism is also available by clicking on the desired entities in the Display window with the Control key down.

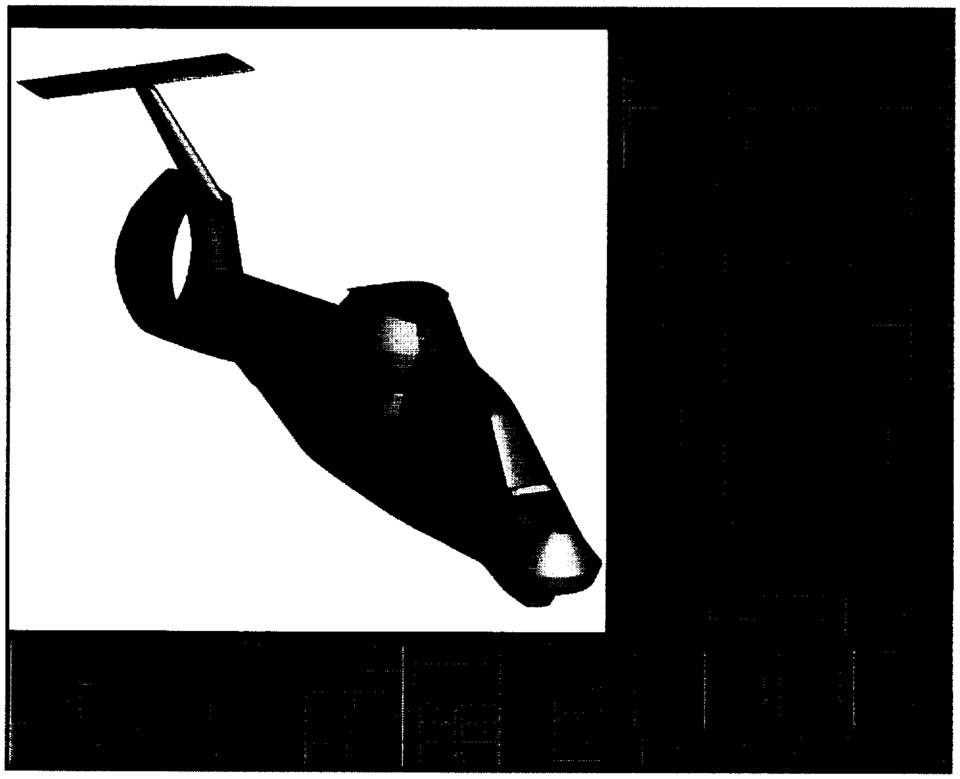

Figure 2: OVERGRID main windows: upper left - Display, lower left - Controls, upper right - Main, lower right - Selection.

3. General Grid Tools

General grid tools in OVERGRID fall into two classes: diagnostic tools for analyzing entity attributes, and manipulation tools for modifying entities. These are described in more details below.

3.1. Diagnostic Tools

The Controls window offers widgets for toggling the display of entity attributes such as tangent and normal vectors for surfaces, entity identification numbers, and iblank information as illustrated in Figure 3. Not typically found in other gridding methods, iblanks are specifically used in overset grids to denote points that are inside or outside of the computational domain. For example, in Figure 3d, grid points in the Cartesian box grid that lie inside the fuselage have an iblank value of 0, while points outside have a non-zero value.

The DIAGS module accessible from the Main Menu window allows the user to check various grid quality functions such as stretching ratios in the J, K, and L directions and truncation error estimates. Wireframe representations of the grid surfaces are colored by the value of the grid quality function. Locations and values of the minimum and maximum are also reported to the user.

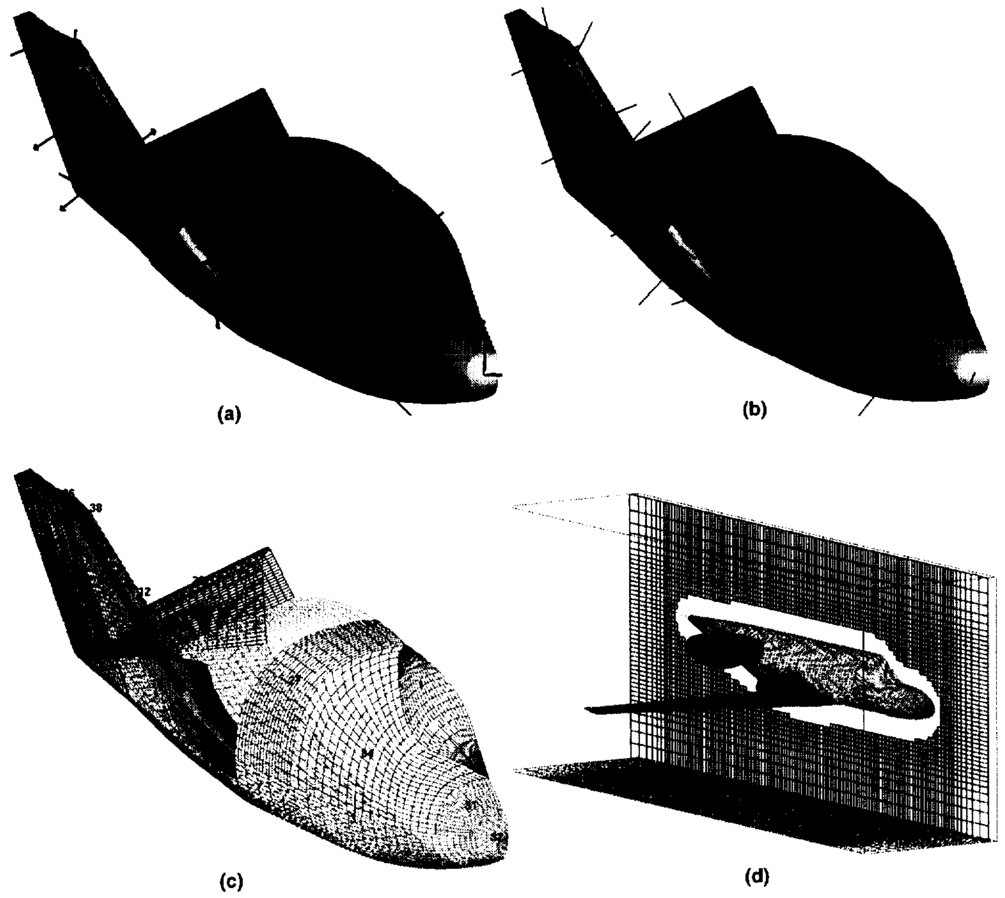

Figure 3: Display of entity attributes in OVERGRID. (a) Surface tangent vectors. (b) Surface normal vectors. (c) Entity identification numbers. (d) Iblank information.

3.2. Manipulation Tools

Three modules are available in OVERGRID for general grid manipulation.

GRIDED - grid editing tool

The GRIDED module provides the following list of commonly used functions for modifying a grid.

(1) Swap J and K, K and L, or J and L grid indices.
(2) Reverse one or more of J, K and L indices.
(3) Mirror about $X = 0$, $Y = 0$, or $Z = 0$ plane.
(4) Scale or translate.

(5) Rotate about X, Y, or Z Cartesian axes.
(6) Extract a subset.
(7) Extrapolate in one or more of $\pm J$, $\pm K$, $\pm L$ directions by any number of layers using a specified stretching ratio.
(8) Concatenate any two volume, surface or curve grids in the specified direction (J, K, or L).
(9) Split an entity into two along a constant J, K, or L direction at a specified index.
(10) Automatically concatenate any number of surface or curve grids in arbitrary relative orientations. A tolerance parameter allows adjacent grids with small gaps in between to be concatenated.
(11) Smooth any subset of a grid in one or more directions in J, K, and L.

SRAP - grid redistribution tool

The SRAP module is used to redistribute grid points on a surface (or curve) entity which is treated as a collection of surface curves in J and K (J only for a curve entity). Points along each surface curve in the J and/or K direction are fitted to a cubic spline, and then redistributed based on user input specifications. There is an option to project the new points back onto the original piece-wise linear definition of each surface curve. Redistribution can occur in any number of segments in each direction where each segment is defined by a start and end index. The user has three input specification options for redistributing points in a segment.

(1) Specify the new number of points and end grid spacings (OVERGRID reports the maximum stretching ratio).
(2) Specify the maximum stretching ratio and end grid spacings (OVERGRID reports the new number of points needed).
(3) Specify a uniform spacing (OVERGRID reports the new number of points needed so that the grid spacing in a uniform mesh will not exceed the specified spacing).

The input grid spacings can be in absolute units or relative to the total arc length of the segment. The current absolute end grid spacings of the segment are displayed as information for the user. Results of redistribution options (2) and (3) are shown in Figure 4.

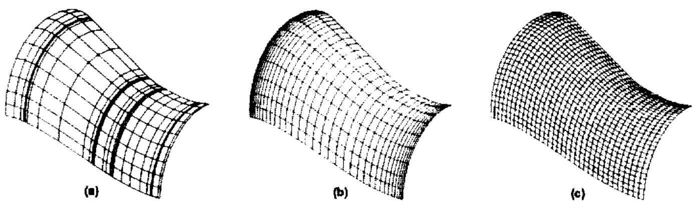

Figure 4: SRAP functions in OVERGRID. (a) Original surface. (b) Redistribution in one direction with clustering at end points. (c) Redistribution to uniform spacing in both directions.

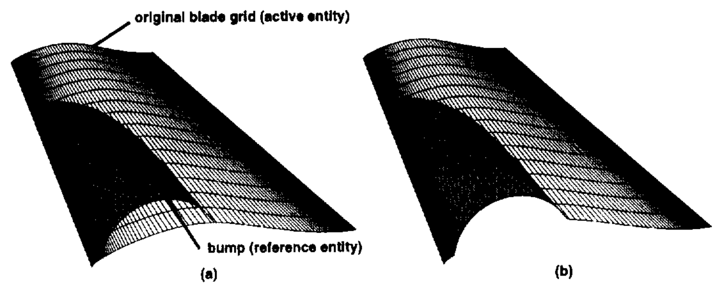

Figure 5: PROGRD function in OVERGRID. (a) Original reference entity (bump grid) and active entity (blade grid). (b) Active entity (blade grid) after projection.

PROGRD - grid projection tool

The PROGRD module in OVERGRID offers a subset of the capabilities of the batch mode module with the same name in Chimera Grid Tools. In OVERGRID, the user defines a reference entity set consisting of one or more surfaces, and an active entity set consisting of surfaces and/or curves. Entities in the active set are projected onto the bilinear surface representation of the entities in the reference set. The projection can be performed in the surface normal direction of the reference entities, or in the X, Y, or Z directions. After the projection, the maximum distance moved by a point in the active entity set is reported to the user. The user can choose from one of the following 3 options if an active point falls outside of the reference surfaces.

(1) Do not move point.
(2) Project point to closest cell on reference surfaces.
(3) Extrapolate reference surfaces tangentially and project point to the extrapolated surfaces.

Figure 5 illustrates the use of PROGRD to model a bump on a blade. A surface grid was originally generated on a clean blade. Subsequent design changes introduced a small bump on the blade. The user decided that it is not critical to model the bump/blade intersection line exactly, and that keeping the final configuration in a single grid is more important. PROGRD is used for this task by projecting the original blade grid (active entity) onto the bump grid (reference entity). Points on the blade grid outside of the bump are undisturbed by selecting option 1 above.

4. Overset Grid Tools

A current strategy for creating overset grids around a complex configuration consists of the following steps.

(1) A high fidelity multiple panel network definition of the surface geometry is obtained. Conversion from other data types may be necessary.
(2) Seam curves are extracted from the surface geometry. These curves are where a grid line should be placed, e.g., intersection curve between components, sharp surface discontinuities, high surface curvature contours, and open boundaries.

(3) The surface geometry is decomposed into four sided domains. Some are bounded by one or more seam curves while others are not bounded by any seam curves. Concatenation and splitting of seam curves may be necessary.
(4) Surface grids are generated on the decomposed domains by hyperbolic or algebraic methods.
(5) Body-conforming volume grids in the near field are created from the surface grids by hyperbolic marching.
(6) Stretched Cartesian box grids are generated to enclose the near-field volume grids and to extend the computational domain to the far field.
(7) Connectivity between the volume grids is established using domain connectivity software.[24–27]

To the frustration of overset grid users for many years, most common grid generation packages do not contain tools that are convenient for performing the above steps. OVERGRID was specifically designed to address steps 2 to 6 above, taking full advantage of the freedom allowed by the overset approach to grid generation. Implementation of the various overset related tools in OVERGRID are discussed in the subsections below. In each case, a batch mode code with the same name also exists in the Chimera Grid Tools package.

4.1. Seam Grid Preparation Tools

The SEAMCR[15] module is used to automatically extract seam curves from a multiple panel network definition of the surface geometry, and to automatically create surface grids around the surface discontinuities. Only the seam curve extraction mode of SEAMCR is discussed below. Details on the algorithm for the automatic surface grid creation mode are given in Ref. 15.

In OVERGRID, seam curves are automatically extracted by SEAMCR based on a single user-specified parameter: an angle threshold. If the angle between the surface normals of any two neighboring quadrilateral cells on the surface panel networks is greater than the angle threshold, the edge separating the two cells is selected. All such edges are concatenated automatically in an appropriate manner to form seam curves. Certain rough or under-resolved regions of the surface may produce extranneous seam curves. These can be removed interactively by the user in OVERGRID. Figure 6 shows seam curves automatically extracted by SEAMCR for the V-22 tiltrotor fuselage, wing and tail. Some extranneous seam curves are produced near the junction between the horizontal tail and the fuselage due to the given poor local surface geometry resolution.

4.2. Surface Grid Generation Tools

After the seam curves have been created, the user has to determine a decomposition of the surface geometry into domains suitable for surface grid generation. The decomposition process frequently results in domains bounded by only one seam curve. Since neighboring grids are allowed to overlap arbitrarily, the other three boundaries of these domains can be freely floated. Such requirements are ideally suited for hyperbolic surface grid generation. In cases where two or more seam curves bound a

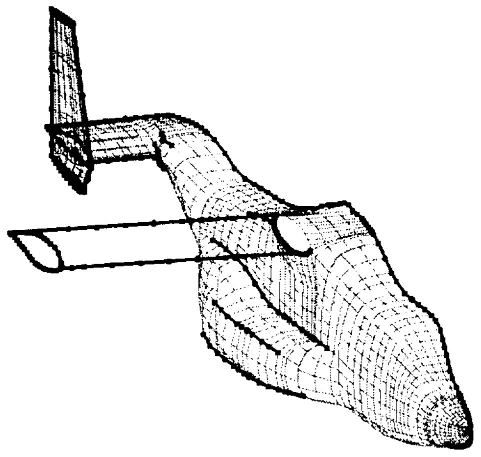

Figure 6: Seam curves produced by the SEAMCR module for the V-22 fuselage and wing. Surface panels for the wing are not shown.

domain, algebraic methods are more appropriate. The SURGRD module described in Section 4.2.1 below has been designed for the above gridding strategy.

After creating surface grids in domains bounded by one or more seam curves, there may be regions of the surface that have not been covered yet. The user can define more initial curves and then use SURGRD to create more surface grids to fill the gaps; or use the SBLOCK module described in Section 4.2.2 below to fill the gaps with automatically generated overlapping algebraic grids. In certain applications, it is desirable to create a wake cut behind a wing-type geometry to form a C grid. The WKCUT module discussed in Section 4.2.3 below has been designed for this purpose.

4.2.1. SURGRD - hyperbolic/algebraic surface grid generator

The SURGRD[20] module is used to create surface grids from 1, 2, 3, or 4 initial curves, using hyperbolic marching, algebraic marching, or transfinite interpolation (TFI) methods. Surface grids are created to conform to a bilinear representation of the surface geometry defined by a collection of panel networks.

For domains bounded by one initial curve, hyperbolic or algebraic marching is used. In most situations, hyperbolic marching is selected to provide orthogonality for the grid lines emanating from the initial curve. However, algebraic marching is sometimes more suitable to create skewed grid lines due to geometric constraints, e.g. marching from a wing/body intersection curve onto a swept wing by following a family of isoparametric lines on the wing surface geometry definition.

For both hyperbolic and algebraic marching, OVERGRID provides widgets to specify the marching distance, initial and/or end spacings, and either the number of points to use or the maximum stretching ratio. Also, for hyperbolic marching, limited control of grid lines emanating from the end points of the initial curve is given via boundary condition specifications, e.g. constant plane, periodic, and others. Smoothing parameters are available for adjustment for very difficult cases, but it is found that modifications of these parameters are not needed over a wide range of

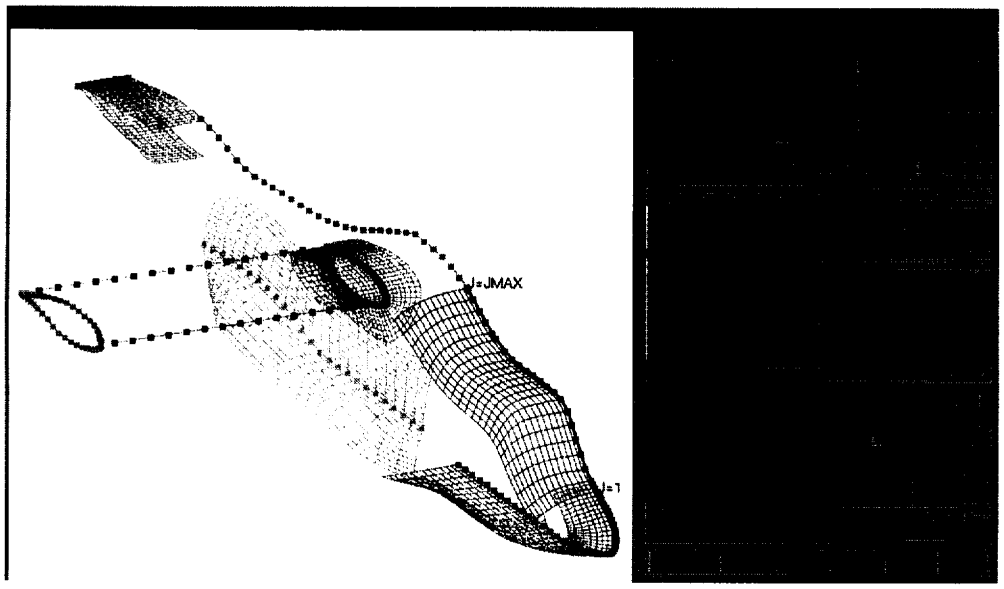

Figure 7: Hyperbolic surface grids created by SURGRD for part of the V-22.

geometries. Figure 7 shows the SURGRD window with widgets for setting input parameters and the DISPLAY window with several V-22 surface grids created by hyperbolic marching.

For domains bounded by two opposite, two adjacent, three, or four initial curves, transfinite interpolation is used. Additional straight lines are automatically constructed by SURGRD for two-curve and three-curve cases to fill in the missing bounding curves. Figure 8 shows the automatically simplified SURGRD window for two opposite initial curves and the DISPLAY window with a TFI grid created between two curves. The SURGRD window simplifies even further for two adjacent, three or four curves since no stretching function need to be specified.

4.2.2. SBLOCK - block surface grid generator

Given the surface geometry and a set of surface grids created around the seam curves, the SBLOCK[14] module is used to automatically generate algebraic surface grids (called block grids by Ref. 14) to fill in regions on the surface not covered by the seam surface grids. Details of the SBLOCK algorithm and code will not be discussed here. The OVERGRID interface is very simple and just provides widgets for the input of the uniform global grid spacing to be used for the algebraic grids.

4.2.3. WKCUT - wake cut surface grid generator

The WKCUT module is used to generate and add a wake cut to the surface grid of a wing to form a C grid. Default parameters are automatically set for the streamwise extent of the wake, the number of points used and the grid spacing. For more difficult cases such as a high wing or low wing, the user can select parameters to modify the deflection angle of the wake cut such that the cut intersects the fuselage. This intersection requirement is needed for the construction of a collar grid [28] in the wing/fuselage junction.

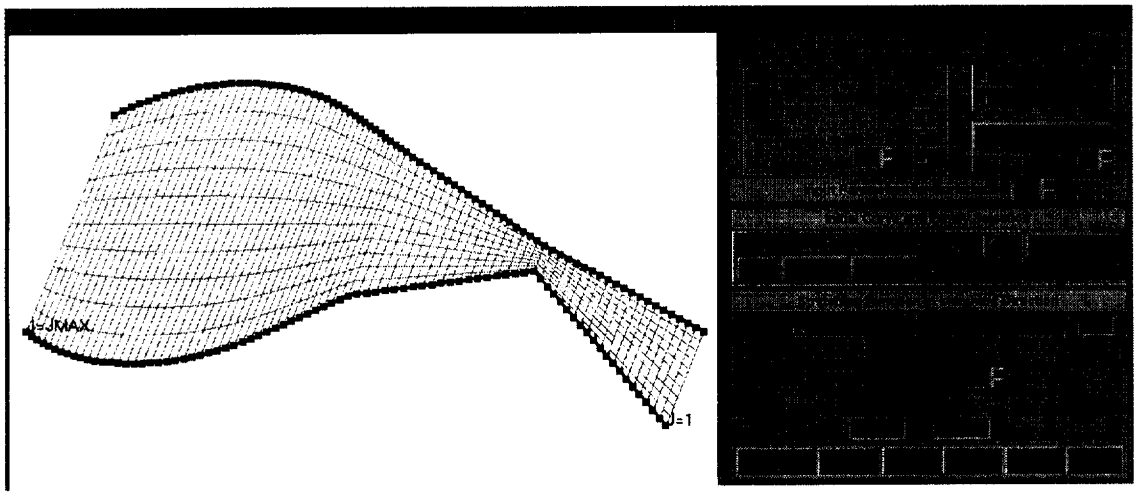

Figure 8: A surface grid created by transfinite interpolation between two opposite curves in SURGRD.

4.3. Volume Grid Generation Tools

After creating a set of overlapping surface grids, body-conforming volume grids have to be generated. Again, the overset approach only requires neighboring volume grids to overlap. This allows the specification of just the surface grid while the other five faces of the volume domain are free to float. A hyperbolic marching scheme is particularly suited for this type of grid. Significant time savings over elliptic methods is possible in using a marching scheme instead of an iterative scheme, and only one instead of six faces need to be defined. Moreover, hyperbolic methods naturally provides the tight clustering needed near the surface for viscous computations, as well as high quality nearly orthogonal grids everywhere. The HYPGEN module described in Section 4.3.1 has been designed to perform the above gridding scheme.

The body-conforming volume grids are typically grown to a constant distance from the body (e.g., a fraction of the body length so that the outer boundaries of the volume grids are well clear of boundary layer interactions). The BOXGR module described in Section 4.3.2 can then be used to automatically create stretched Cartesian box grids around the near field volume grids and extend the computational domain to the far field.

4.3.1. HYPGEN - hyperbolic field grid generator

The HYPGEN[18,19] module is employed to create a volume grid by marching from a surface grid. For two-dimensional cases, a field grid is generated by marching from a two-dimensional curve. The marching distance, initial and/or end grid spacings and the number of points to use in the marching direction are specified by the user. Boundary conditions are automatically selected by OVERGRID based on the topology of the given surface grid, e.g. periodicity, singular axis point, constant plane, and others. A free floating boundary is selected if no special topology is detected. The user also has the choice to enforce different boundary conditions, and to adjust smoothing parameters if needed for difficult cases. After the volume grid is created, HYPGEN reports if any negative Jacobians are found. This information is also displayed in

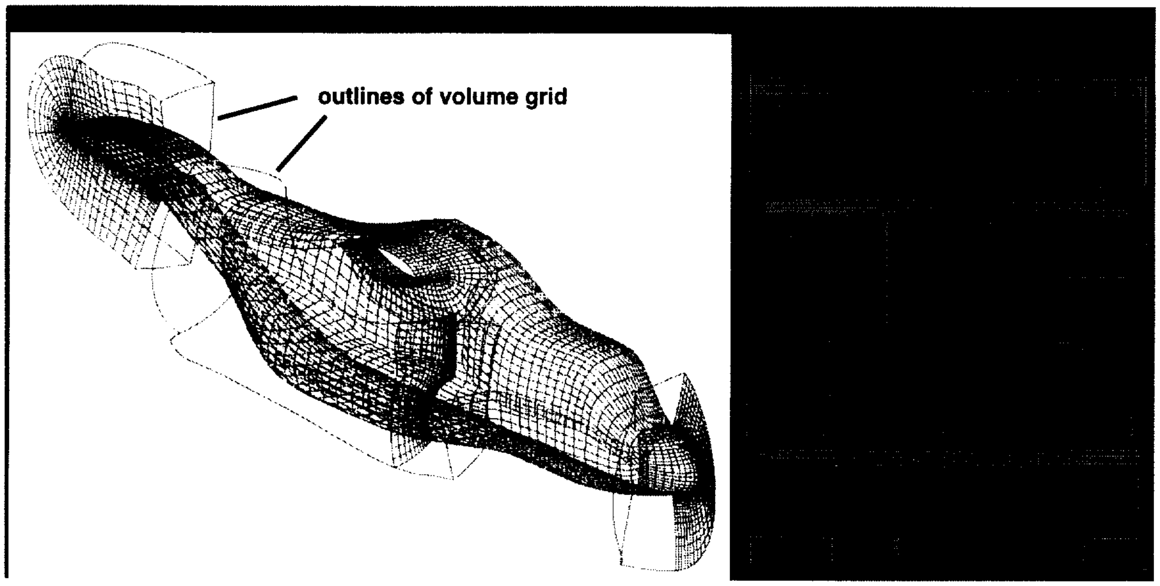

Figure 9: Hyperbolic volume grids created by HYPGEN for part of the V-22.

OVERGRID's HYPGEN window.

Figure 9 shows the HYPGEN window with widgets for setting input parameters, and the DISPLAY window with several volume grids created by hyperbolic marching. For all volume grids, a geometric stretching is used in the normal direction with the same marching distance, initial spacing and number of points. Default boundary conditions are employed for all grids (free floating for the cases shown). The three volume grids shown contain a total of about 25000 points and took a total of less than 15 seconds of user's time (wall clock) to generate on a Silicon Graphics R10000 workstation.

4.3.2. BOXGR - stretched Cartesian grid generator

The BOXGR module is used to create a Cartesian box grid consisting of an interior core with uniform spacing, and optional stretched outer layers in the plus and minus X, Y, and Z directions. In BOXGR's automatic mode, the user selects one or more volume grids, and BOXGR will automatically create a Cartesian box grid with uniform spacing that completely encloses all the selected volume grids (see Figure 10a). The uniform spacing is automatically chosen to match the average grid spacing at the outer boundaries of the volume grids. In BOXGR's manual mode, the coordinates of the corners of the interior core can be explicitly prescribed. In both automatic and manual mode, extra stretched layers in all directions can easily be added by specifying a distance and a stretching ratio (see Figure 10b).

The computational domain can be extended to the far field via the stretched layers, or via the ellipsoidal shell option in BOXGR. In the latter, BOXGR automatically generates an ellipsoidal surface grid that fits one or more cells inside the outer boundaries of a given stretched Cartesian box grid, thus providing proper overlap; and the grid spacings in the tangential direction are made to match those of the Cartesian grid (see Figure 10c). A hyperbolic volume grid can then be grown from this sur-

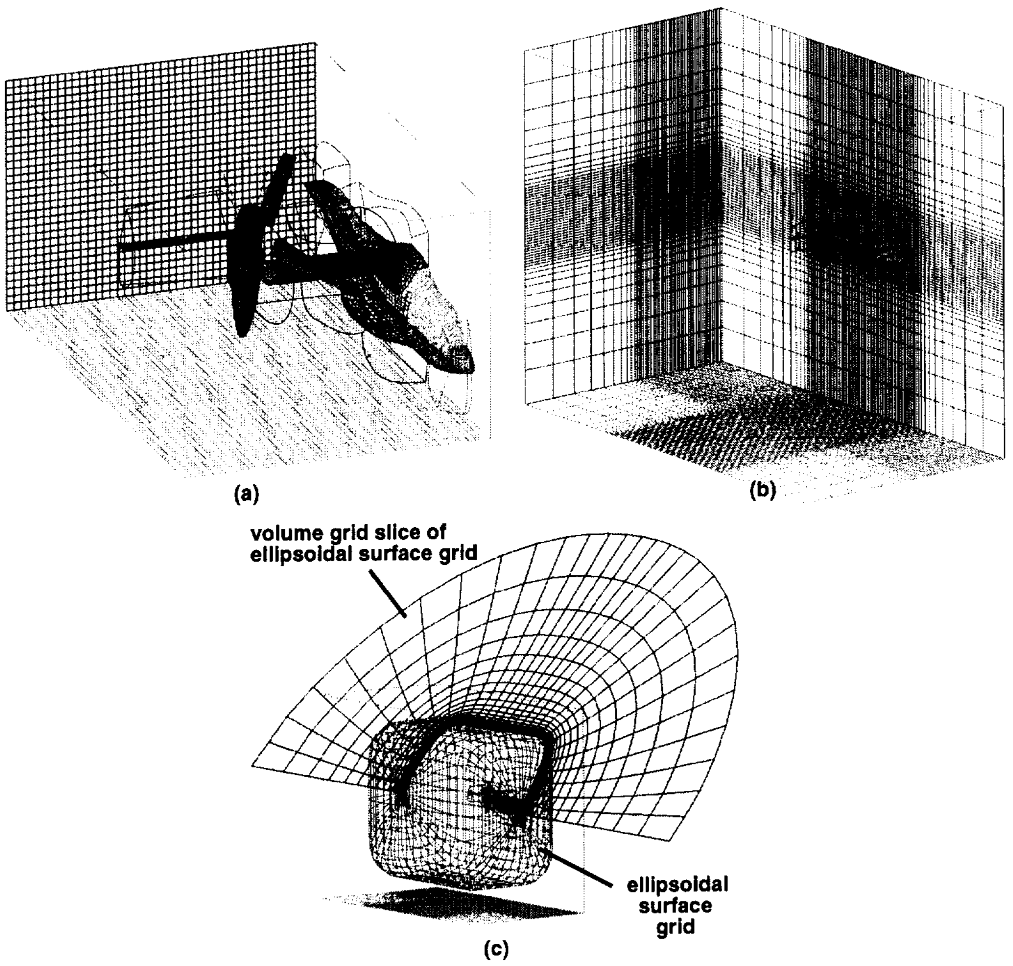

Figure 10: (a) Cartesian box grid with uniform core created automatically by BOXGR. (b) Cartesian box grid with extra stretched layers. (c) Ellipsoidal surface grid created by BOXGR and volume grid created by HYPGEN.

face to take the computational domain to the far field. The ellipsoid topology allows a uniform expansion of the grid, resulting in grid point savings over the stretched Cartesian box option which keeps the tight uniform core spacing out to the far field.

5. Conclusions and Future Plans

A unified and practical graphical user interface for efficient and robust generation of overset grids is presented. Tools implemented in the interface follow a gridding strategy which is also outlined in the paper. It is the author's belief that this is the first time that such an interface has been specifically designed for overset grids. OVERGRID has been distributed to users in various government laboratories, industry groups, and universities under a non-disclosure agreement. Feedback from users indicate a typical reduction in grid generation time by a factor of 3 to 4.

Future plans for OVERGRID include the development of capabilities to read other common forms of surface geometry definition such as triangles, NURBS surfaces, and others; and to create surface grids from such databases. Further development of automated algorithms for creating surface and volume grids is also anticipated.

Acknowledgements

Mouse transforms and direct screen picking of entities in OVERGRID were developed by Steven M. Nash. The batch mode version of the wake cut generation tool WKCUT was created by Dr. Stuart E. Rogers, Steven M. Nash and Dr. Pieter G. Buning. The author would like to thank the many testers of OVERGRID, especially Dr. Robert L. Meakin, Dr. Earl Duque, and Dr. Pieter G. Buning for their patience and constructive comments. This work was performed as part of the DOD High Performance Computing Modernization Program Initiative known as CHSSI, and funded through NASA contract number NAS2-14109, Task 23.

References

1. Steger, J. L., Dougherty, F. C. and Benek, J. A., "A Chimera Grid Scheme," *Advances in Grid Generation*, K. N. Ghia and U. Ghia, eds., ASME FED-Vol. 5, June, 1983.

2. Smith, M., Chawla K., and Van Dalsem, W. R., "Numerical Simulation of a Complete STOVL Aircraft in Ground Effect," AIAA Paper 91-3293, 9th AIAA Applied Aerodynamics Conference, September, 1991.

3. Atwood, C. A. and Van Dalsem, W. R., "Flowfield Simulation about the SOFIA Airborne Observatory," AIAA Paper 92-0656, January, 1992.

4. Meakin, R., "Moving Body Overset Grid Methods for Complete Aircraft Tiltrotor Simulations," AIAA Paper 93-3350, *Proceedings of the 11th AIAA Computational Fluid Dynamics Conference*, Orlando, Florida, July, 1993.

5. Gomez, R. J. and Ma, E. C., "Validation of a Large Scale Chimera Grid System for the Space Shuttle Launch Vehicle," AIAA Paper 94-1859, *Proceedings of the 12th AIAA Applied Aerodynamics Conference,* Colorado Springs, Colorado, June, 1994.

6. Slotnick, J. P., Kandula, M. and Buning, P. G., "Navier-Stokes Simulation of the Space Shuttle Launch Vehicle Flight Transonic Flowfield Using a Large Scale Chimera Grid System," AIAA Paper 94-1860, *Proceedings of the 12th AIAA Applied Aerodynamics Conference,* Colorado Springs, Colorado, June, 1994.

7. Atwood, C. A., "Computation of a Controlled Store Separation from a Cavity," *J. of Aircraft*, Vol. 32, No. 4, pp.846–852, 1995.

8. Gee, K., Murman, S. M. and Schiff, L. B., "Computation of F/A-18 Tail Buffet," *J. of Aircraft*, Vol. 33, No. 6, pp.1181–1189, 1996.

9. Srinivasan, G.R. and Klotz, S.P., "Features of Cavity Flow and Acoustics of the Stratospheric Observatory For Infrared Astronomy," *Proceedings of the ASME Fluids Engineering Conference,* Vancouver, British Columbia, Canada, June, 1997.

10. Gea, L. M., Halsey, N. D., Intemann, G. A. and Buning, P. G., "Applications of the 3-D Navier-Stokes Code OVERFLOW for Analyzing Propulsion-Airframe Integration Related Issues on Subsonic Transports," ICAS Paper 94-3.7.4, 19th Congress of the International Council of the Aeronautical Sciences, September, 1994.

11. Wai, J., Herling, W. W. and Muilenburg, D. A., "Analysis of a Joined-Wing Configuration," AIAA Paper 94-0657, January, 1994.

12. Duque, E. P. N, Berry J. D., Budge A. M. and Dimanlig A. C. B., "A Comparison of Computed and Experimental Flowfields of the RAH-66 Helicopter," Proceedings of the 1995 American Helicopter Society Aeromechanics Specialist Meeting, Fairfield County, Connecticut, 1995.

13. Rogers, S. E., Cao, H. V. and Su, T. Y., "Grid Generation for Complex High-Lift Configurations," AIAA Paper 98-3011, 29th AIAA Fluid Dynamics Conference, Albuquerque, New Mexico, June, 1998.

14. Chan, W. M. and Meakin, R. L., "Advances Towards Automatic Surface Domain Decomposition and Grid Generation for Overset Grids," AIAA Paper 97-1979, *Proceedings of the AIAA 13th Computational Fluid Dynamics Conference,* Snowmass, Colorado, 1997.

15. Chan, W. M. and Gomez, R. J., "Advances in Automatic Overset Grid Generation Around Surface Discontinuities," submitted to the AIAA 14th Computational Fluid Dynamics Conference, Norfolk, Virginia, June, 1999.

16. Chawner, J. R. and Steinbrenner, J. P., "Automatic Structured Grid Generation Using GRIDGEN (Some Restrictions Apply)," *Proceedings of NASA Workshop on Surface Modeling, Grid Generation, and Related Issues in Computational Fluid Dynamics (CFD) Solutions,* NASA CP 3291, May, 1995.

17. Akdag, V. and Wulf, A., "Tuned Grid Generation with ICEMCFD," *Proceedings of NASA Workshop on Surface Modeling, Grid Generation, and Related Issues in Computational Fluid Dynamics (CFD) Solutions,* NASA CP 3291, May, 1995.

18. Chan, W. M. and Steger, J. L., "Enhancements of a Three-Dimensional Hyperbolic Grid Generation Scheme," *Appl. Math. & Comput.* Vol. 51, pp.181–205, 1992.

19. Chan, W. M., Chiu, I. T. and Buning, P. G., "User's Manual for the HYPGEN Hyperbolic Grid Generator and the HGUI Graphical User Interface," NASA TM 108791, October, 1993.

20. Chan, W. M. and Buning, P. G., "Surface Grid Generation Methods for Overset Grids," *Computers and Fluids*, Vol. 24, No. 5, pp.509–522, 1995.

21. Ousterhout, J. K., *Tcl and the Tk Toolkit*, Addison-Wesley, 1994.

22. Welch, B., *Practical Programming in Tcl and Tk*, 2nd ed., Prentice Hall, 1997.

23. Walatka, P. P., Buning, P. G., Pierce, L. and Elson, P. A., "PLOT3D User's Manual," NASA TM 101067, 1990.

24. Suhs, N. E. and Tramel, R. W., "PEGSUS 4.0 User's Manual," AEDC-TR-91-8, AEDC/PA, Arnold Air Force Base, Tennessee, 1991.

25. Meakin, R. L., "A New Method for Establishing Intergrid Communication Among Systems of Overset Grids," AIAA Paper 91-1586, *Proceedings of the 10th AIAA Computational Fluid Dynamics Conference,* Honolulu, Hawaii, 1991.

26. Maple, R. C. and Belk, D. M., "Automated Set Up of Blocked, Patched and Embedded Grids in the Beggar Flow Solver," *Numerical Grid Generation in*

Computational Fluid Dynamics and Related Fields, Ed. N. P. Weatherill et al., Pine Ridge Press, pp.305–314, 1994.

27. Henshaw, W. D., Chesshire, G. and Henderson, M. E., "On Constructing Three Dimensional Overlapping Grids with CMPGRD," *Software Systems for Surface Modeling and Grid Generation*, Ed. Robert E. Smith, Hampton, VA, Langley Research Center, NASA CP 3143, pp.415–434, 1992.

28. Parks, S. J., Buning, P. G., Steger, J. L. and Chan, W. M., "Collar Grids for Intersecting Geometric Components Within the Chimera Overlapped Grid Scheme," AIAA Paper 91-1587, *Proceedings of the 10th AIAA Computational Fluid Dynamics Conference*, Honolulu, Hawaii, 1991.